Ramdeobaba University, Nagpur

Department of Computer Science and Engineering

Session: 2025-26

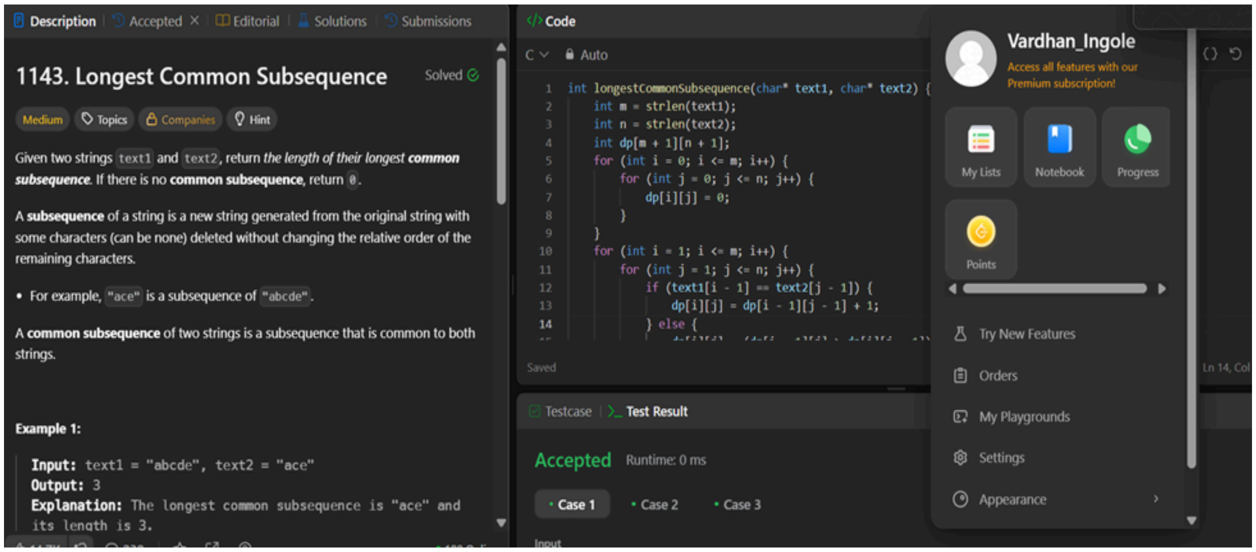
DAA LAB **III Semester**

Name: Vardhan Ingole
Section: A4
Batch: B3
Roll No: 43

Practical -5 (LeetCode)

LeetCode Assessment:

https://leetcode.com/problems/longest-common-subsequence/submissions/1784499446/



1143. Longest Common Subsequence

Given two strings `text1` and `text2`, return *the length of their longest* ***common subsequence***. If there is no **common subsequence**, return `0`.

A **subsequence** of a string is a new string generated from the original string with some characters (can be none) deleted without changing the relative order of the remaining characters.

- For example, `"ace"` is a subsequence of `"abcde"`.

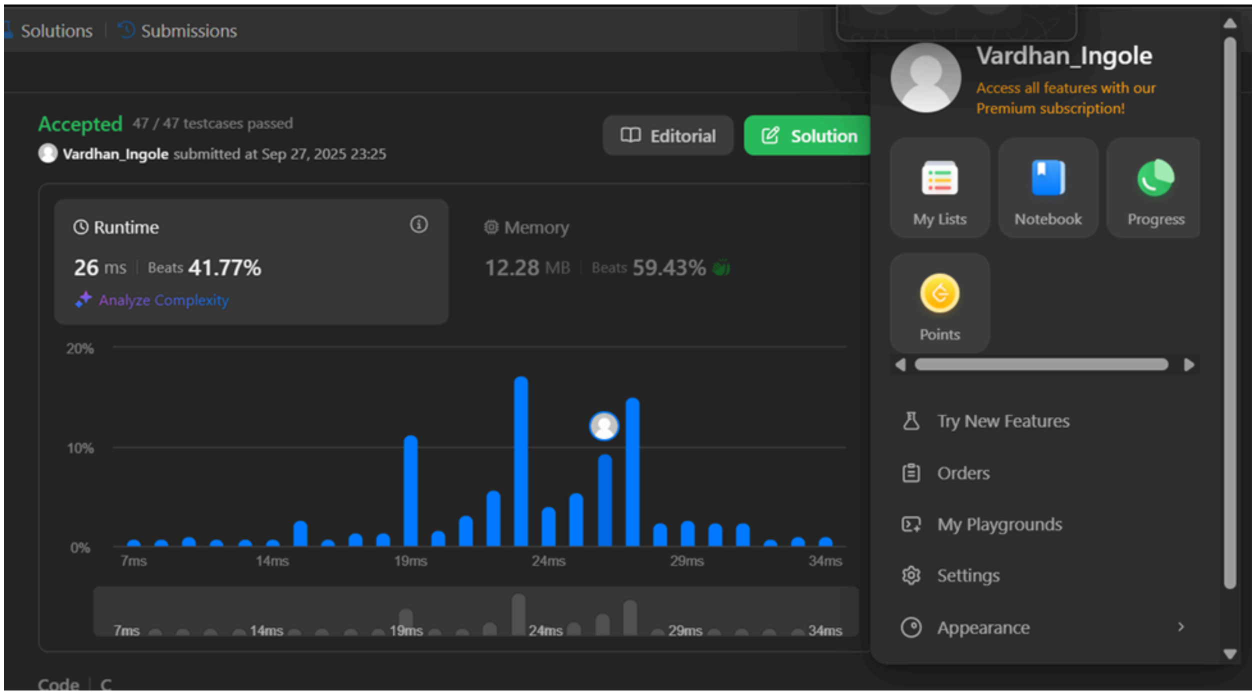
A **common subsequence** of two strings is a subsequence that is common to both strings.

Example 1:

```
Input: text1 = "abcde", text2 = "ace"
Output: 3
Explanation: The longest common subsequence is "ace" and
its length is 3.
```

```c
int longestCommonSubsequence(char* text1, char* text2) {
    int m = strlen(text1);
    int n = strlen(text2);
    int dp[m + 1][n + 1];
    for (int i = 0; i <= m; i++) {
        for (int j = 0; j <= n; j++) {
            dp[i][j] = 0;
        }
    }
    for (int i = 1; i <= m; i++) {
        for (int j = 1; j <= n; j++) {
            if (text1[i - 1] == text2[j - 1]) {
                dp[i][j] = dp[i - 1][j - 1] + 1;
            } else {
```

Accepted Runtime: 0 ms



Accepted 47 / 47 testcases passed

Vardhan_Ingole submitted at Sep 27, 2025 23:25

Runtime

26 ms | Beats 41.77%

Analyze Complexity

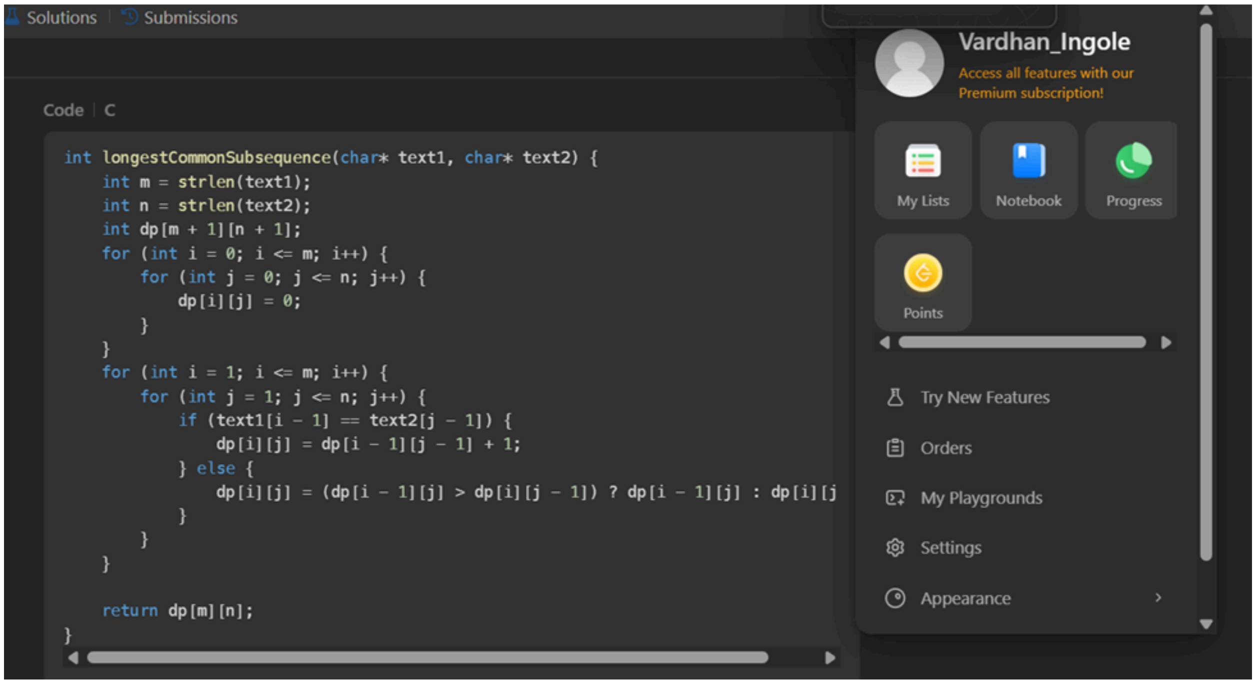
Memory

12.28 MB | Beats 59.43%

Vardhan_Ingole

Access all features with our Premium subscription!

Code | C



Code | C

```c
int longestCommonSubsequence(char* text1, char* text2) {
    int m = strlen(text1);
    int n = strlen(text2);
    int dp[m + 1][n + 1];
    for (int i = 0; i <= m; i++) {
        for (int j = 0; j <= n; j++) {
            dp[i][j] = 0;
        }
    }
    for (int i = 1; i <= m; i++) {
        for (int j = 1; j <= n; j++) {
            if (text1[i - 1] == text2[j - 1]) {
                dp[i][j] = dp[i - 1][j - 1] + 1;
            } else {
                dp[i][j] = (dp[i - 1][j] > dp[i][j - 1]) ? dp[i - 1][j] : dp[i][j
            }
        }
    }

    return dp[m][n];
}
```